



# 06. Arrays (vectores y matrices)

Programación - 1º DAM

Luis del Moral Martínez

versión 20.10

Bajo licencia CC BY-NC-SA 4.0



# Contenidos del tema

1. Vectores (arrays)
2. Matrices (arrays multidimensionales)

# 1. Vectores (arrays)

## Concepto de array

- Un array, o vector, es una **sucesión** de objetos del mismo tipo
- Los **elementos** se llaman elementos del array y se numeran de forma consecutiva (0,1,2...)
- Normalmente, un array se usa para almacenar elementos de tipo **int**, **char**, **float**...
- El array puede ser creado también de estructuras o de objetos
- El array tiene una **longitud** determinada

45	17	34	91	29	63
0	1	2	3	4	5

# 1. Vectores (arrays)

## Declaración de un array

- Un array debe ser declarado antes de poder utilizarlo
- A la hora de declararlo debemos indicar el **tamaño del array** (número de elementos)
- El número de elementos se indica entre corchetes (**[** y **]**)

**tipo\_dato** nombreArray **[numeroElementos];**

**int** notas **[10];**



[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

# 1. Vectores (arrays)

## Acceso a los elementos de un array

- Para acceder a los elementos necesitamos un **subíndice**
- El subíndice es de tipo **entero**
- El primer elemento del array tiene de índice 0, el siguiente 1...

<b>int</b> notas[10];	10	9	7	6	9	5	3	2	7	5	notas [5]	5
	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]		[5]

- Podemos imprimir el contenido de un vector o asignarlo a otra variable **cout << notas[5];**
- Cuidado: **C++ no comprobará si el índice está en el rango correcto (0...9)**

# 1. Vectores (arrays)

## Almacenamiento de un array en memoria

- Los elementos de los arrays se almacenan en bloques contiguos de memoria

<b>int</b> edades [5];	
edades [0]	a
[1]	34.36
[2]	ff3434ds
[3]	34
[4]	-15

# 1. Vectores (arrays)

## Inicialización de un array (1)

- Antes de utilizar los elementos de un array hay que inicializarlos
- Cuidado: es posible que en esas celdas de memoria pueda haber “basura”

<b>int</b>	<b>edades</b>	<b>[5];</b>	
	<b>edades</b>	[0]	a
		[1]	34.36
		[2]	ff3434ds
		[3]	34
		[4]	-15

# 1. Vectores (arrays)

## Inicialización de un array (2)

- Podemos inicializar un array en el momento de su declaración o usando un bucle
- En el momento de su **declaración**:
  - Si lo inicializamos en su declaración, no es necesario indicar el tamaño máximo (es opcional)

```
int notas [10] = {4, 5, 6, 7, 8, 10, 4, 7, 2, 9};
```

```
char nombre [ ] = { 'L' , 'U' , 'i' , 's' };
```

- Usando un bucle:

```
for (int i = 0; i < longitudVector; i++)  
    notas[ i ] = 5;
```

- Abre el fichero **06\_01\_arrays\_1.cpp**



# 1. Vectores (arrays)

## Arrays de caracteres y cadenas de texto

- Hasta ahora hemos trabajado con el tipo **char**, que representa un carácter
- Cuando en **cout** usamos las comillas dobles ("), estamos imprimiendo una **cadena de texto**
- Las cadenas de texto (también llamadas **cadenas de caracteres**) son agrupaciones de char
- Las cadenas de caracteres tienen un **inicio** y un **final**
- El final de la cadena está indicado por el **carácter nulo** '**\0**'

**char** nombre[10];

M	a	r	g	a	r	i	t	a	'\0'
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

## 2. Matrices (arrays multidimensionales)

### Concepto de array multidimensional

- Un **array multidimensional** tiene más de una dimensión
- En consecuencia, tiene más de un **índice**
- Se pueden crear tantas dimensiones como se necesite, pero lo normal son 2 (**matrices**)
- Un array de **2 dimensiones** equivale a una tabla con **filas** y **columnas**

**int** notas[2][10];

2 dimensiones

2 filas

10 columnas

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
[0]	1	2	7	2	9	5	8	6	3	5
[1]	4	5	2	7	9	5	2	8	9	10

## 2. Matrices (arrays multidimensionales)

### Declaración de arrays multidimensionales

- Los arrays multidimensionales se declaran de la siguiente forma:

**tipo\_dato** nombreArray [numeroFilas][numeroColumnas];  
**int** notas [2][10];

- Esta declaración implica crear dos arrays
  - Un array de 2 elementos y cada elemento es, a su vez, un array de 10 elementos

**int** notas[2][10];

2 dimensiones

2 filas

10 columnas

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
[0]	1	2	7	2	9	5	8	6	3	5
[1]	4	5	2	7	9	5	2	8	9	10

## 2. Matrices (arrays multidimensionales)

### Inicialización de arrays multidimensionales

- Los arrays multidimensionales pueden inicializarse mediante un bucle
- También puede inicializarse en el momento de su declaración:

```
int notas [2][5] = {  
    {0, 1, 2, 3, 4},  
    {5, 6, 7, 8, 9}  
};
```

## 2. Matrices (arrays multidimensionales)

### Almacenamiento de un array multidimensional en memoria

- Los elementos de los arrays se almacenan en bloques contiguos de memoria

**int** edades [2][3];

[0][0]
[0][1]
[0][2]
[1][0]
[1][1]
[1][2]

## 2. Matrices (arrays multidimensionales)

### Acceso a los elementos de un array multidimensional

- Para acceder a los elementos necesitamos dos **subíndice**
- Los subíndices son de tipo **entero**
- El primer elemento del array tiene de índice 0, el siguiente 1...

**int** notas[2][10];

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
[0]	1	2	7	2	9	5	8	6	3	5
[1]	4	5	2	7	9	5	2	8	9	10

- Podemos imprimir el contenido de un vector o asignarlo a otra variable `cout << notas[1][5];`
- Cuidado: **C++ no comprobará si el índice está en el rango correcto (0...9)**

## 2. Matrices (arrays multidimensionales)

### Acceso a los elementos de un array multidimensional

- Para acceder a los elementos necesitamos dos **subíndice**
- Los subíndices son de tipo **entero**
- El primer elemento del array tiene de índice 0, el siguiente 1...

**int** notas[2][10];

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
[0]	1	2	7	2	9	5	8	6	3	5
[1]	4	5	2	7	9	5	2	8	9	10

notas[1][5]

	[5]
[1]	5

- Podemos imprimir el contenido de un vector o asignarlo a otra variable `cout << notas[1][5];`
- Cuidado: **C++ no comprobará si el índice está en el rango correcto (0...9)**

## 2. Matrices (arrays multidimensionales)

### Acceso a un array multidimensional usando bucles

- Podemos acceder a un array multidimensional usando bucles anidados

```
for (int i = 0; i < 2; i++)  
    for (int j = 0; j < 10; j++)  
        notas[ i ][ j ] = 5;
```

- Abre el fichero **06\_02\_arrays\_2.cpp**



## 2. Matrices (arrays multidimensionales)

### Consideraciones finales

- Los arrays pueden ser pasados como **argumento** a una función
- Los arrays en C++ son pasados **por referencia** (esto lo estudiaremos más adelante)
- Dentro de una función, si se modifica un array, se está modificando el array original
- Al pasar un array como argumento a una función:
  - Tenemos que indicar las dimensiones del array en alguna variable adicional
  - Si no las indicamos, corremos el riesgo de “salirnos del array” (**violación de segmento**)

# Créditos de las imágenes y figuras

## Cliparts e iconos

- **Obtenidos mediante la herramienta web [IconFinder](#)** (según sus disposiciones):
  - Diapositiva 1
  - Según la plataforma IconFinder, dicho material puede usarse libremente (free comercial use)
  - A fecha de edición de este material, todos los cliparts son free for comercial use (sin restricciones)

## Resto de diagramas y gráficas

- Se han desarrollado en PowerPoint y se han incrustado en esta presentación
- Todos estos materiales se han desarrollado por el autor
  - Si se ha empleado algún icono externo, este se rige según lo expresado anteriormente